Beggars and philanthropists β7

User manual

PETER BØGH ANDERSEN, DEPARTMENT OF INFORMATION AND MEDIA STUDIES, UNIVERSITY OF AARHUS

Table of contents

Table of contents 2
1 Introduction 3
1.1 Revisions 3
2 Basic behavior 4
2.1 Beggars 4
2.2 Philanthropists 6
2.3 Simulations 7
3 Collection of statistics 8
3.1 Histograms 8
3.2 Catastrophes 10
3.3 Trend curves 11
4 Collection of statistics 12
5 Windows 12
5.1 Control window 12
5.2 Birds eye 14
5.3 Parameters 15
5.4 Histograms 17
5.5 Trend curves 18
6 Parameter space exploration 21
7 Files 23
7.1 Save exceldata 23
7.2 Save beggars 25
7.3 Save Reportdata 26
8 Modules 6
9 References 7

1 Introduction

This manual describes a simulator tailored to investigate the budgeting model described in Prange & Serritslew 2005, Prange (2006).

It requires the Python interpreter plus its standard modules. It was developed in Python 2.4.4 under Windows 2000 and XP 2002.

The system is run by clicking the *view* module. The system is a *beta*-version which means that it probably contains errors.

1.1 Revisions

The following revisions were agreed on a seminar May 15-16, 2008.

1 We must be able to save a particular parameter-setting on a file and read it into the system again since it is inconvenient to reset parameters for each session (Section 5.3).

2 The beggar's distance to a philanthropists should be measured from the beggar's current philanthropist, not from his own position. The purpose is to make the spatial metaphor more consistent.

3 It must be possible to cancel collection of the individual statistics in order to avoid excessive memory-loads with long simulations. The trend-curves must be excepted since we need the current average beggar budget when we generate new beggars. Section 5.1.

4 There should be no catastrophe detection in the beggar-life-span statistics. It is time-consuming and is not used for anything.

5 We introduce a new trend diagram, *numOfCatas***,** that shows the number of beggars experiencing catastrophes. Section 5.5.

6 Log-log plots should use the absolute values. Not implemented yet.

7 Beggar-states should be saved as decimal numbers since reducing small numbers to zero will exclude them from being treated in the statistics package. Principle: use integers for humans, but decimals for machines.

8 Beggar-states should contain the following information: *beggarname, iteration, budget, grant, distance, arrivals to the current philanthropist, departures from the current philanthropist, selfMoved* (did the beggar move?), *selfReviewed* (was the beggar reviewed?), *appReviewed* (the number beggars reviewed by the current philanthropist), *numOfComp* (the number of competitors the beggar could see), *catastrophe* (did his grant suffer a catastrophe?), *coApplicants* (the actual number of co-applicants to the current philanthropist) , *deaths* (the number of beggars going bankrupt at the current philanthropist), *simulationNo*. When we run a parameter exploration, the values for the chosen parameters should be appended. Section 7.2.

9 Add a window where a space of parameters can be explored. Simulations are run for each combination of parameter values. The parameters are picked from a listbox. The parameter space can itself be saved and used again. It is possible to specify 4 parameter values in 4 parameters. If we use all 16 possibilities, the system will gener-

ate the following amount of numbers: 16 × iteration number × number of agents. If iteration number = 800 and number of agents = 30, it will generate 384.000 beggar-states.

10 All combination scatter-diagrams should be removed.

11 The following parameters should be removed from the parameter window: *y-dimension, Beggar/Phil ratio, Phil vision, Survival Threshold, selectPhil2*

2 Basic behavior

The model consists of a grid on which a number of agents are placed. A position can only contain one agent. There are two kinds of agents: *philanthropists* have money they can spend on grants to *beggars*. The positions and distances of the grid represent professional or social ties. The longer the distance between a beggar and a philanthropist is, the looser the ties are.

2.1 Beggars

The behavior of beggars is as follows:

```
MakeApplication
 find philanthropists in the vicinity you can see
 if there are philanthropists
  calculate your competitors
  calculate the distance to the philanthropist
  calculate the successrate of the philanthropist
  select the most promising philanthropist
  if there is one
   if the selected one is not the same you used before
    move to him
   calculate a budget proposal
   send the proposal to the selected philanthropist

HandleAnswer
 if you did not received a grant
  go bankrupt
 else:
  if you can survive with the grant received
   spend the money
  else
   go bankrupt
```

Beggars are initialized as follows: the budget of a newborn beggar is calculated as a Gaussian function of the mean beggar budget from the previous iteration and a variance which is 1/3 of the mean. In iteration 0 the mean is set to *Mean Init Begg* which defaults to 19. The balance is set to survivalThreshold*initialBudget where survivalThreshold = 0.05.

Each time a beggar has received a grant, the following happens:

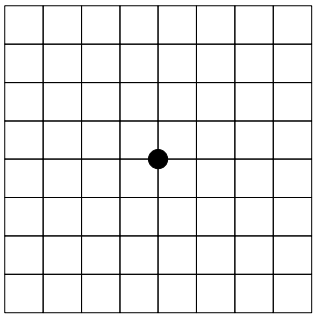
- A new budget proposal is generated based on the old one: `budgetProposal = oldbudget + gauss(oldbudget*p_muIncrBegg, p_sigmaIncrBegg)` with defaults p_muIncrBegg = 0.02, p_sigmaIncrBegg = 0.001.
- An expenditure is generated based on the old budget: `expenditure = oldbudget * gauss(p_muExpenBegg, p_sigmaExpenBegg)` with defaults p_muExpenBegg = 0.95, p_sigmaExpenBegg = 0.002

- A new balance is calculated based on the old balance, the grant and the expenditure: newbalance = oldbalance + grant – expenditure.

If the balance is positive, the beggar survives and the budget proposal and the new balance becomes the old budget and old balance, respectively.

However, before he can get a grant, he must find a philanthropist.

First the beggar looks around in the vicinity he can see, and collects a list of funding opportunities. The vicinity is determined by the beggar's view which is set to 4 as default. He can see 4 positions away from himself. Prange & Serritslew (2005) says that with a vision of 4, the beggar can see the 8 * 8 = 64 surrounding squares. However, in the implementation agents are positioned at grid-corners, so the beggar with a vision of 4 can inspect 9*9 = 81 positions (if we exclude himself, it is 80).



The rectangle defining the view of the beggar in the middle if his vision = 4

The algorithm is:

```
findPhilantropists
 determine the neighborhood you can see
 find the philanthropists in this neighborhood
 if there are any:
  for all philanthropists in the neighborhood:
   get the closest empty position in the vicinity of 3
    of the philanthropist
   if there is one:
    record the philanthropist as a funding opportunity
```

There is thus a small chance that every position around a philanthropist is occupied. The philanthropist is therefore not eligible.

A funding opportunity consists of a philanthropist, his success rate, competitors around the philanthropist, the length of the path to him and the vacant location. The competitors are calculated as follows:

```
calculateCompetitors
 for all opportunities:
  for all beggars in the vicinity of 3 of the philanthropist except me
   if the beggar has not chosen a philanthropist
      or the beggars philanthropist = the philanthropist
   add the beggar to the competitorlist
  store the sum of the expected budgets of competitors
```

The reason why we do not consider beggars that have chosen another philanthropist than the one we are looking at is that such beggars are not likely to change philanthropist. This situation can occur if two philanthropists A and B are very close. B's beggars may be in the vicinity of A but will probably stick to B.

The path is calculated as the distance to the philanthropist. However, the distance to the philanthropist from the previous round is set to 0. The distance to a new philanthropist is a cost that can result in reduced grants, but the distance to the old philanthropist does not incur costs.

If the beggar has already an old philanthropist, the distance is measured from him, otherwise from the beggar himself.

```
calculatePath
 for all opportunities
  if the philanthropist is the one selected in the previous round
   the path is 0
  else
   the path is the distance to the philanthropist
```

The success rate is calculated as the sum of the grants divided by the sum of the applications.

If a philanthropist has not enough money for all applications, some of the beggars are reviewed and risk getting cut down.

The review probability R is calculated as (`p_Base.value =` 0.25 `= base risk`):

```
R = baserisk + (1 - baserisk)*float(dist/sqrt(2*n*n))
```

where n is the dimension of the grid, dist is the distance to the philanthropist. Sqrt(2*n*n) is the diagonal of the grid, i.e. the maximal distance The probability is between 0.25 and 1.00. If the distance = 0 the probability is 0.25. If the distance is equal to the diagonal, the probability is 1.

The philanthropist is now selected. First the ones with the largest expected payoff are selected. Among these, the nearest are selected.

The selection method is defined as follows: let bComp = sum of competitors budget, bPhil = budget of the philanthropist, newBudget = own new budget, and baseRisk = p_Base = 0.25. Then the algorithm is:

```
selectPhilantropist2
if there are funding opportunities:
 for all opportunities:
  if bComp + own budget is not greater the philanthropists budget:
   the expected payoff is own budget
  else
   expected payoff = (1-R)* newBudget +
    (R*(bPhil - (1 - baseRisk)*o.bComp)*newBudget)
     /(baseRisk*(bComp) + newBudget)
 calculate the largest expected payoff of all opportunities
 find all opportunities where expected payoff = the largest one
 among these, choose the nearest one as the selected opportunity
else
 record that no opportunity was selected
```

2.2 Philanthropists

The behavior of the philanthropists is as follows: philanthropists start with a budget determined by `gauss(p_muInitPhil, p_sigmaInitPhil)`. Defaults: p_muInitPhil = 80, p_sigmaInitPhil = 20. The new budget of the philanthropist is calculated from the old one:

```
Replenish budget
 Newbudget = oldbudget + gauss(float(oldbudget*p_muIncrPhil),
          p_sigmaIncrPhil)
```

with the defaults p_muIncrPhil = 0.001 and p_sigmaIncrPhil = 0.02

Applications are handled and grants are sent:

```
Handle applications:
if you received applications:
 calculate grants
 send grants
```

The grants is calculated in the following way. We assume that

- Budgetlist = the list of applications
- Bbrev = the sum of the budgets of the reviewed beggars
- Bbadj = the sum of the budgets of all applicants
- Excess = bbadj – philanthropist's new budget. The sum applied for which exceeds the philanthropist's budget

Then we do the following:

```
calculateGrantList:
 if no applications were received:
  the grant list is empty
 else
  bbadj = the sum of beggar budgets
  if bbadj <= own budget:
   the grant list = the budget list
   successrate = 1
  else:
   bbrev = 0
   while bbrev < excess:
    for all applications:
     if it happens to be reviewed:
      put it on the review list
      bbrev += the amount of the application
  reduce all reviewed budgets by (newBudget – bbadj + bbrev)/ bbrev
  successrate = bbadj – the sum of the grants
```

The algorithm 'squeezes' the reviewed applications into the budget part *philanthropist budget – sum of reviewed budgets.*

2.3 Simulations

One iteration of a simulation is executed as follows:

```
Generate new statistics objects
For all philanthropists:
 Replenish budget
For all beggars:
 make applications
For all philanthropists:
 handle applications
For all beggars:
 handle answers
Remove bankrupt beggars
Update statistics
Add new beggars
```

The number of new beggars at the end of a simulation is calculated as follows:

```
newBeggars = (sum of philanthropists budgets – sum of surviving beggars'
budgets)/ average beggar budget
```

Where

```
average beggar budget = sum of beggar budgets / the number of beggars in
the beginning of the simulation)
```

3 Collection of statistics

The simulation collects two types of statistics, histograms and trend curves.

3.1 Histograms

Histograms record the number of occurrences of measurements within an interval.

The following data are collected in a simulation

- *beggarLifeSpan:* x-axis: intervals of life span, y-axis: number of beggars with the life span
- *grantDifferences;* x-axis: intervals of grant differences, new grant – old grant, y axis: number of occurrences of the grant difference
- *Applicants:* x-axis: intervals of philanthropist budgets, y-axis: number of beggars applying to a philanthropist with the budget.

There is a parameter *Collect stat from* that can be set and default is 5. It hinders collection of histogram statistics in the first iterations whose number is less than its value. The reason is that the initial rounds may be atypical.

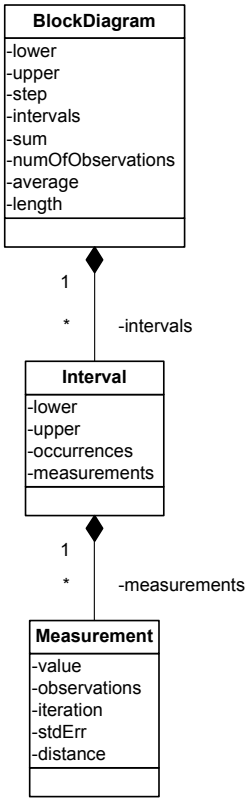
Histograms contain a list of intervals, defined by a lower and upper bound. Measurements equal to or higher than the lower bound but lower than the upper bound +1 are recorded in the interval. The histogram is initialized with a set of intervals. If measurements outside the intervals are encountered, new intervals are added dynamically.

The histogram keeps track of the current list of intervals via the variables *lower* and *upper*. Lower is equal to the lower bound of the lowest interval, and upper is equal to the lower bound of the highest interval.

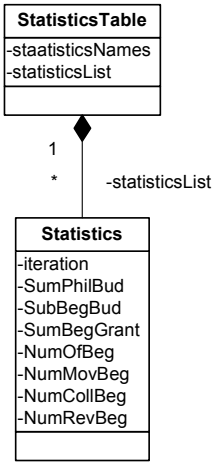
When a new value is encountered, it is recorded by adding a new Measurement to the Interval. A measurement contains the value recorded, the number of observations of this value, the standard error at the time of recording, and the distance to the mean measured in standard errors.

In most of the cases, there is only one observation of the value. However, the window “applicants” shows the number of beggars (y-axis) applying for a certain philanthropist budget (x-axis). In this case, there are as many observations as there are beggars applying for this particular budget.

The variable *numOfObservations* in the histograms sums the total number of observations. The variable *sum* sums the total number of values observed. The *average* is the sum divided by the number of observations.

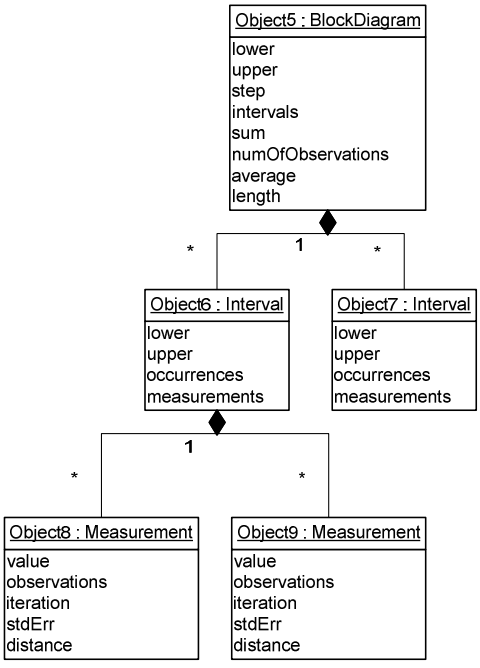


UML class diagram of histograms



UML class diagram of trend curves

During a simulation, the following structure is built:



Data structure built during execution

3.2 Catastrophes

It is possible to set a catastrophe alarm for grant differences. The catastrophe alarm makes a sound and writes a message to the console if the new value is a certain distance, measured in standard errors, from the mean. The catastrophe is also recorded in the beggar states. The distance defaults to 4 but can be changed in the parameter window. Since the standard error needs 30 observations to make sense, it is only calculated after 30 observations.

The standard error is calculated as follows:

$$SE = \frac{\sqrt{\sum_{i=1}^{n} (Y_i - Aver_n)^2}}{\sqrt{n}}$$

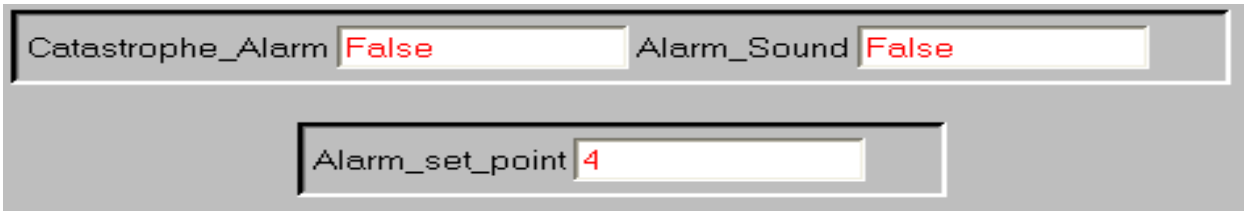
Where *n* is the number of observations, *Y* is an observed value, and *average* is the mean of all observed values at the time of calculation.

```
standardError:
 if numOfObservations ≠ 0 and  numOfObservations ≠ 1:
  result = 0
  for i in the intervals:
   for v in the interval's measurements:
    add the square of measurement's difference from the average
      to the result
  take the square root of the result
  divide this by the square root of the number of observations
  return the result
 else:
  return None
```

Each time a value is recorded, the standard error and the distance of the value to the mean in terms of standard errors are calculated. If the distance is larger than the setpoint, the system beeps, writes a message to the console, and registers a catastrophe. Running the algorithm with a normal distribution gives from 0 to 3-4 of catastrophes per 400 observations. Running it with grant differences gives a lot more:

Catastrophe in Grant differences. Value = 11.8237450858, average = 0.0358989656981, standard Error = 1.77679220774, distance = 6.63434140965, number of observations = 43
Catastrophe in Grant differences. Value = 23.8390114703, average = 0.448911543609, standard Error = 2.84925101702, distance = 8.20920999485, number of observations = 66
Catastrophe in Grant differences. Value = 18.8759282103, average = 0.700159760307, standard Error = 3.99544304585, distance = 4.54912465061, number of observations = 152
Catastrophe in Grant differences. Value = 17.0615345915, average = 0.685296104611, standard Error = 4.02921820396, distance = 4.06437121494, number of observations = 175
Catastrophe in Grant differences. Value = 24.1064031003, average = 0.724121667706, standard Error = 4.12579783802, distance = 5.66733571314, number of observations = 191

The catastrophe detector is only active for *grantDifferences.* 3 parameters can be set for the catastrophe alarm:



Parameters for the catastrophe alarm.

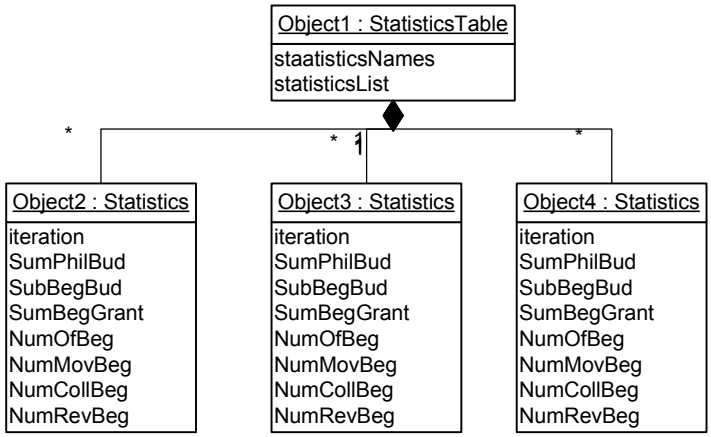
Catastrophe Alarm switches the alarm on and off. Write False or True in the text field. *Alarm Sound* switches the sound on and off. Write False or True in the text field. The *Alarm Set Point* changes the limit for catastrophes. Write an integer in the field. In all cases, end with a *return*.

3.3 Trend curves

Trend curves measure a number of properties for each iteration and stores them with the iteration number. The properties are:

SumPhilBud	The sum of philanthropists' budgets in an iteration
SumBegBud	The sum of beggars' budgets in an iteration
SumBegGra	The sum of beggars' grants in an iteration
NumOfBeg	The number of surviving beggars after an iteration
NumMovBeg	The number of moving beggars in an iteration
NumCollBeg	The number of collapsing beggars in an iteration
NumRevBeg	The number of reviewed beggars in an iteration
AveBudBeg	The average budget of beggars in an iteration
AveBudPhil	The average budget of philanthropists in an iteration
numOfCatas	The number of catastrophes in an iteration

During a simulation, a list of statistics is built up:



Data structure built up for trend curves. Each object represent one iteration where figures for SumPhilBud etc. are recorded.

4 Collection of statistics

The statistics are collected in the following way:

Name	Point of measurement	Comment
grant differences	Beggar's *handleAnswer* method just after the new grant has been recorded	'No grant' is recorded as a 0
beggarLifeSpan	Beggars' *goBankRupt* method	The life span of a beggar is death iteration – birth iteration
Applicants	Philanthropists' *calculateGrant* method	The philantropists' budget is the measurement. The number of applications received received from beggars is the number of observations.
SumPhilBud	Philantrophists' *replenishBudget* method	
SumBegBud	Beggars' *makeApplication* method	
SumBegGra	Beggars' *handleAnswer* method	
NumOfBeg	The method *runOneSimulation* after the iteration has ended.	The variable measures the number of surviving beggars after iteration i.
NumMovBeg	Beggar's *move* method	
NumCollBeg	Beggar's *goBankRupt* method	
NumRevBeg	Philantrophists' *calculateGrantList* method	The variable counts the number of beggars selected for review
AveBudBeg	The method *runOneSimulation* after the iteration has ended.	The variable counts the average budget of beggars
AveBudPhil	The method *runOneSimulation* after the iteration has ended.	The variable counts the average budget of philanthropists
numOfCatas	.The beggar's *handleAnswer* method	The variable counts the number of catastrophes in an iteration

5 Windows

The system contains the following types of windows:

- The control window
- The birds eye window
- The parameter window
- Histograms
- Trend curve diagrams

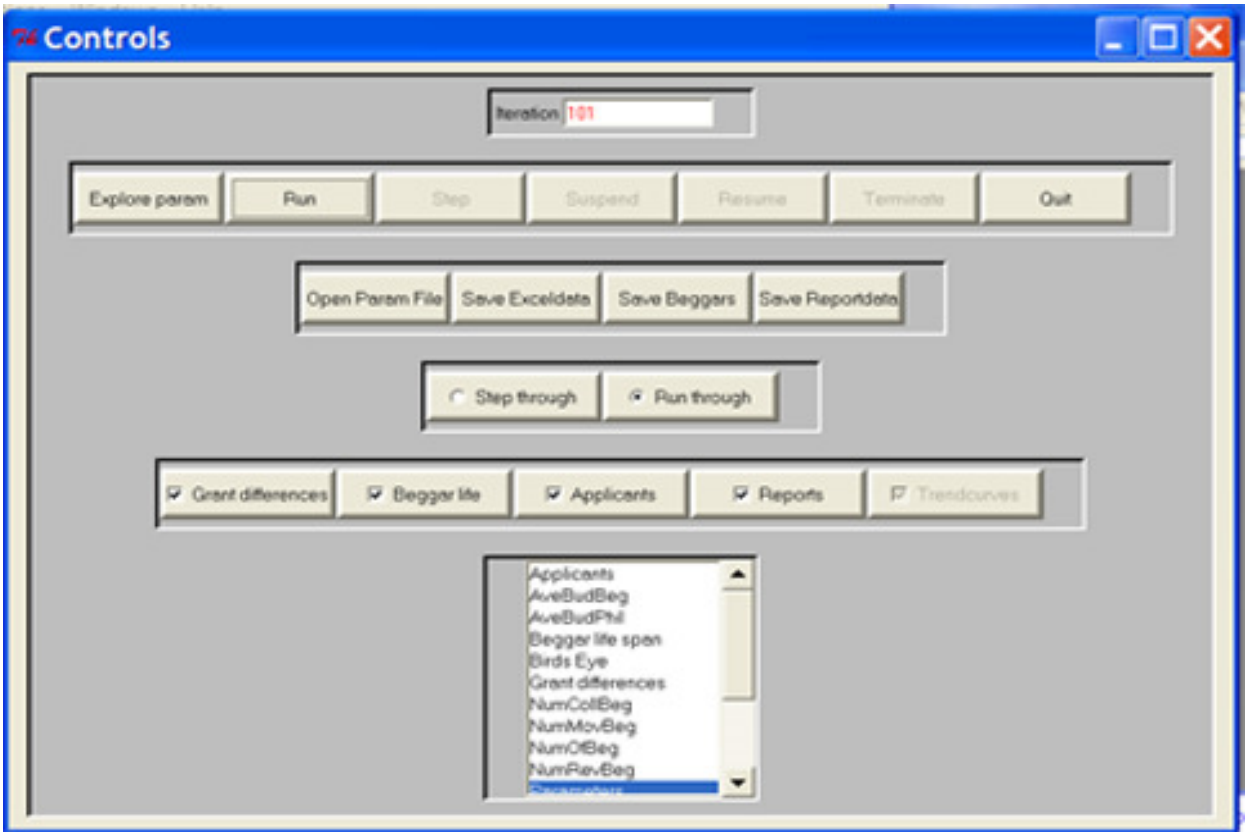
5.1 Control window

The control window is used to control the simulation. It can run in two modes, *step through* and *run through*. The former allows the user to manually step through the simulation by clicking the *Step* button. The latter runs automatically through the simulation.

The *Terminate* button terminates the current simulation after the current iteration. The *Suspend* button suspends it after the current iteration; the simulation can be resumed by pressing the *Resume* button. The program is terminated by pressing *Quit*.

Note that a simulation can only be suspended and terminated between iterations.

The button 'Explore Param' opens a window for exploration of a parameter space (see Section 6).



The control window

The button *Open Param File* enables the user to use a parameter setting he has previously saved to file from the *parameter* window.

There are three buttons for saving data to file:

- *Save exceldata:* saves 4 comma delimited files: *Trend curves* (stores the trend curves), *Lifespan* (a histogram of the beggars lifespan), *Grant Differences* (a histogram of grant differences), *Applicants* (the number of beggars applying for a given philanthropist budget)
- *Save Beggars:* saves 1 comma delimited file, *BeggarStates*. It records the budget, grant and the distances from the mean measured in standard errors for a beggar in an iteration.
- *Save Reportdata:* saves a detailed verbal report, *Report*, of the simulation.

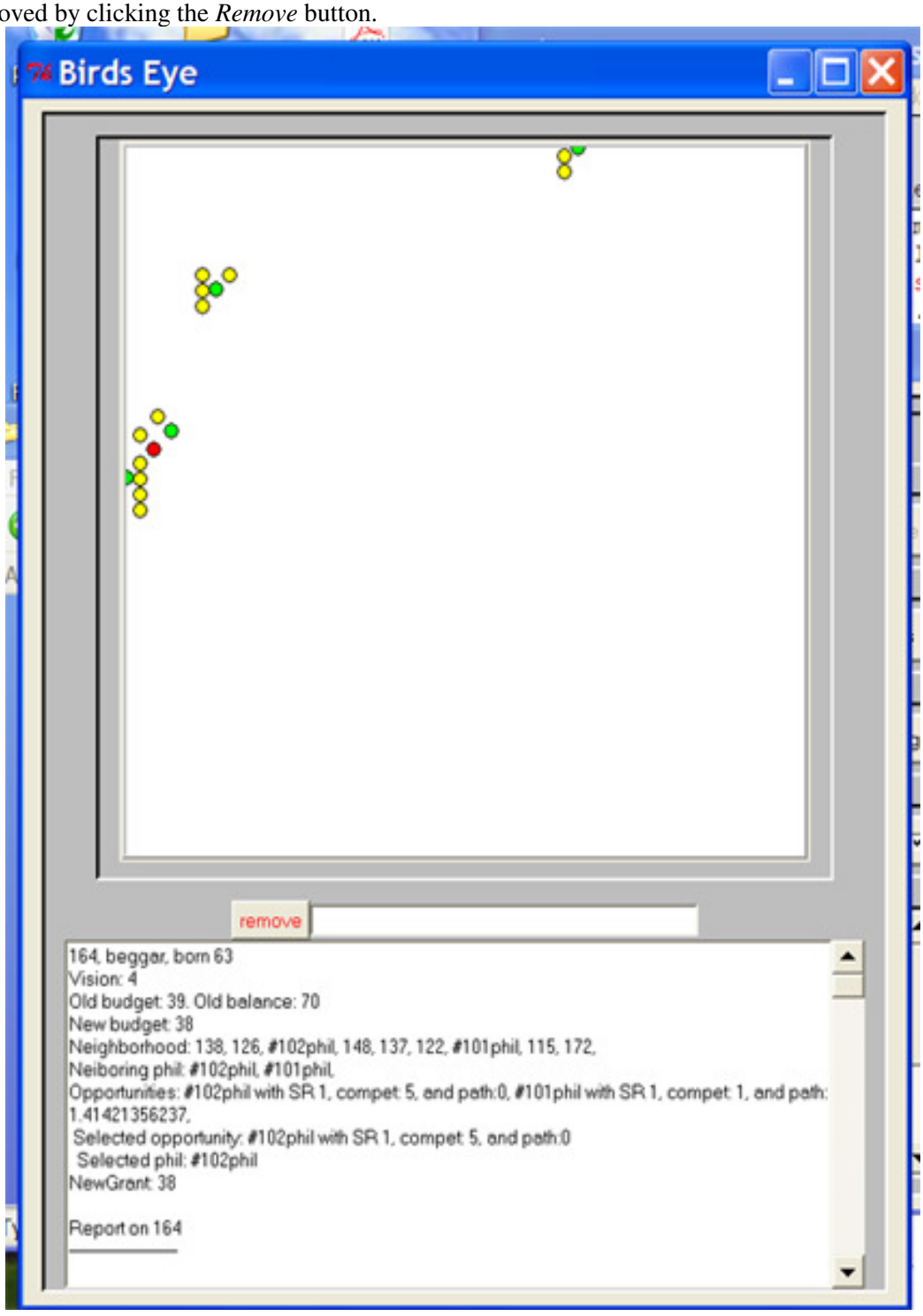
The five check-boxes make it possible to switch off collection of statistics (Section 3):

- Grant differences: histogram of grant differences
- Beggar life: histogram of life durations
- Applicants: histogram of the number of applicants to philanthropists
- Reports: a textual narrative of events occurring during the simulations

Finally, there is a menu of windows. A window can be opened by clicking at its name in the scroll-field. Don't open the same window twice. Each window displays a statistical measurement.

5.2 Birds eye

This window shows an animation of the current iteration. Beggars are yellow, philanthropists are green. Moving the cursor over an agent reveals his name. Clicking an agent opens a window with his current state and a textual history of his life. The window can be removed by clicking the *Remove* button.


Birds Eye
remove
164, beggar, born 63
Vision: 4
Old budget: 39. Old balance: 70
New budget: 38
Neighborhood: 138, 126, #102phil, 148, 137, 122, #101phil, 115, 172,
Neiboring phil: #102phil, #101phil,
Opportunities: #102phil with SR 1, compet: 5, and path:0, #101phil with SR 1, compet: 1, and path:
1.41421356237,
Selected opportunity: #102phil with SR 1, compet: 5, and path:0
Selected phil: #102phil
NewGrant: 38
Report on 164

The birds eye view

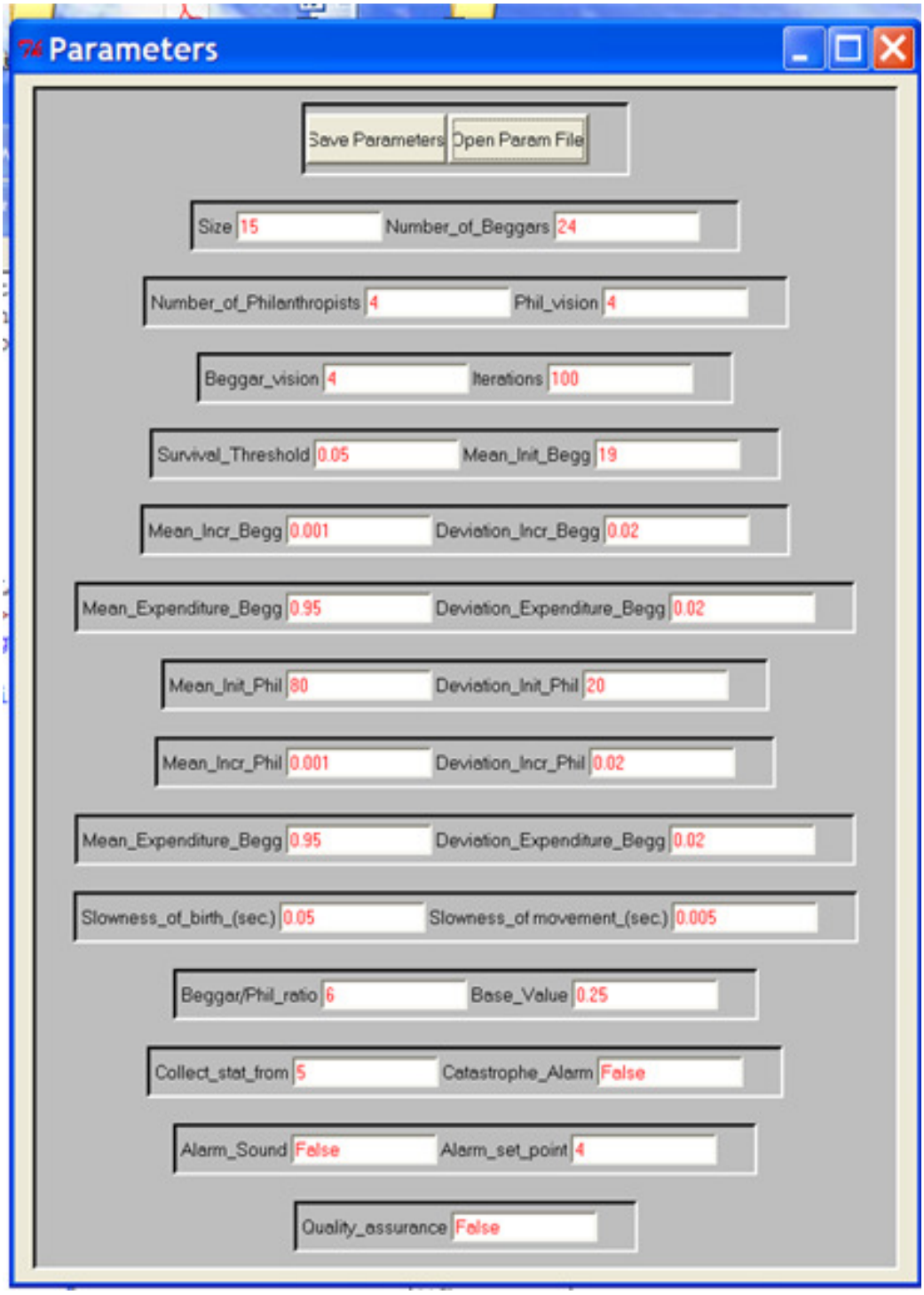
Example of textual history:

```
164, beggar, born 63
Vision: 4
Old budget: 39. Old balance: 70
New budget: 38
Neighborhood: 138, 126, #102phil, 148, 137, 122, #101phil, 115, 172,
Neiboring phil: #102phil, #101phil,
Opportunities: #102phil with SR 1, compet: 5, and path:0, #101phil with SR
1, compet: 1, and path:1.41421356237,
 Selected opportunity: #102phil with SR 1, compet: 5, and path:0
  Selected phil: #102phil
NewGrant: 38

Report on 164
--------------------

*************
164 was born at iteration 63


*************
Iteration 64, found 2 philantropists, calculated 3(#102phil) and
2(#101phil) competitors, calculated paths to 2 philantropists - 3
(#102phil) and 4 (#101phil), calculated the successrates 1(#102phil) and
1(#101phil), selected philantropist #102phil, calculated a budget proposal
of 30 crowns, moved from Nothing to #102phil, sent proposal for 30 crowns
to #102phil, received 30 crowns from #102phil, handled answer, spent 28
crowns

*************
Iteration 65, found 2 philantropists, calculated 2(#102phil) and
2(#101phil) competitors, calculated paths to 2 philantropists - 0
(#102phil) and 1 (#101phil), calculated the successrates
0.879512224347(#102phil) and 1(#101phil), selected philantropist #102phil,
calculated a budget proposal of 30 crowns, sent proposal for 30 crowns to
#102phil, received 30 crowns from #102phil, handled answer, spent 28 crowns

…

*************
Iteration 100, found 2 philantropists, calculated 5(#102phil) and
1(#101phil) competitors, calculated paths to 2 philantropists - 0
(#102phil) and 1 (#101phil), calculated the successrates 1(#102phil) and
1(#101phil), selected philantropist #102phil, calculated a budget proposal
of 38 crowns, sent proposal for 38 crowns to #102phil, received 38 crowns
from #102phil, handled answer, spent 37 crowns, moved from #102phil to
#102phil
```

The birds' eye shows the correct position of the philanthropists. It does not show the real position of the beggars. Instead beggars are moved so close as possible to the philanthropist they have chosen to apply to. The reason is that in some cases, it is difficult to see which beggar belongs to which philanthropist if we use the correct position. There is thus a discrepancy between the underlying model and the view.

5.3 Parameters

The parameter window is used to change various parameters. Type a new value plus a RETURN into a textbox. A parameter setting can be saved (*Save Parameters*) and retrieved again (*Open Param File*).



The parameter window

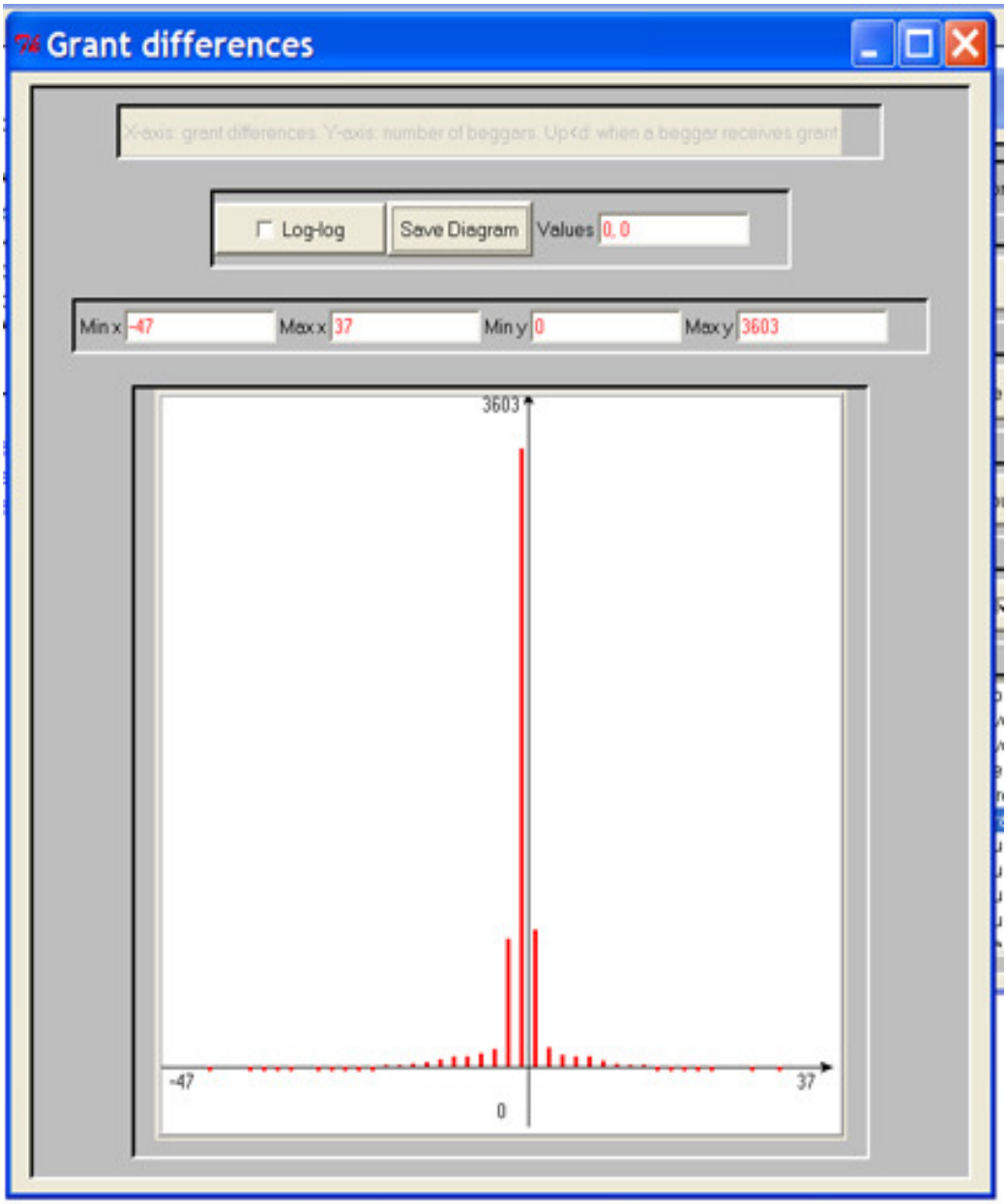
In the table below is listed the name of the parameter, its function, where it is used (*Init* denotes the constructor method for the classes), and its default value.

Name	**Function**	**Used in**	**Default**
size	The dimension of grid	SimulationSpace, init	15
Number of Beggars	Number of beggars at start	SimulationSpace, init	24
Number of Philanthropists	Number of philanthropists at start	SimulationSpace, init	4
Phil vision	The vision of philanthropists	SimulationSpace, init	4
Beggar vision	The vision of beggars	SimulationSpace, init	4

Iterations	The number of iterations run when using the run-mode	SimulationSpace, init	100
Slowness of birth (sec.)	Controls the animation representing the birth of a beggar	BirdsEyeView, addAgentsToDisplay	0.05
Slowness of movement (sec)	Controls the animation representing the movement of a beggar	BirdsEyeView, moveWidget	0.005
Survival Threshold	Controls the initial balance of a beggar.	Beggar, Init	0.05
Mean Init Begg	The mean of the budgets of beggars in the first round	SimulationSpace, init, insertObjects and runOneSimulation	19
Mean Init Phil	The mean of the budgets of philanthropists	initialBudgetPhilantropist in Philanathropist, init	80
Deviation Init Phil	The deviation of the budgets of philanthropists	initialBudgetPhilantropist in Philanathropist, init	20
Mean Incr Begg	The mean increase of the budgets of beggars	newBudgetBeggar in Beggar, makeApplication	0.001
Deviation Incr Begg	The deviation of increase of the budgets of beggars	newBudgetBeggar in Beggar, makeApplication	0.02
Mean Incr Phil	The mean increase of the budgets of philanthropists	newBudgetPhilantropist in Philantrhopist, replenishBudget	0.001
Deviation Incr Phil	The deviation of increase of the budgets of philanthropists	newBudgetPhilantropist in Philantrhopist, replenishBudget	0.02
Mean Expenditure Begg	The mean percentage of the budget that is spent by a beggar	expenditureBeggar in Beggar, spendGrant	0.95
Deviation Expenditure Begg	The deviation of the percentage of the budget that is spent by a beggar	expenditureBeggar in Beggar, spendGrant	0.02
Base Value	The base risk of being reviewed	Beggar, selectPhilantropist2	0.25
Collect stat from	The iteration from which statistics is collected	goBankRupt, Beggar, beggarLifeSpan handleAnswer, Beggar, grantDifferences	5
Catastrophe Alarm	If true, a catastrophe detector is activated in histograms after 30 observations	addValueToStatistics in BlockDiagram	False
Alarm Sound	If true a sound is played for each catastrophe	addValueToStatistics in BlockDiagram	False
Alarm set point	Defines a catastrophe: a distance greater than the set point is a catastrophe	addValueToStatistics in BlockDiagram	4

5.4 Histograms

There are 3 displays of histograms, *Beggar Life Span, Grant Differences,* and *Applicants*.



Grant differences

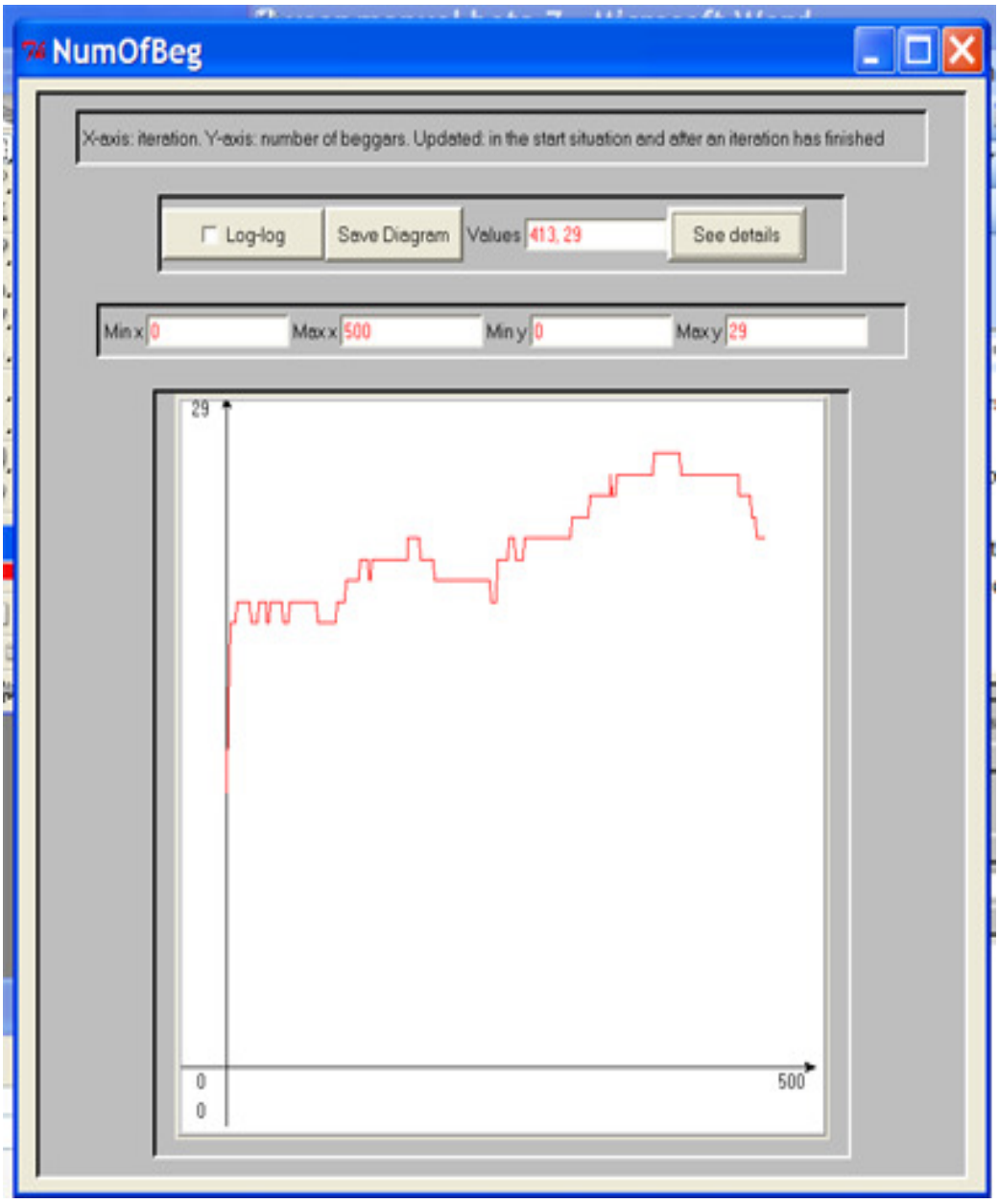
In order to avoid that small numbers disappear on the display, a measurement is displayed 3 pixels below the x-axis, if it was reduced to equal the y-origo during the graphical scaling operations. A negative bar thus means: a small number of occurrences. However, histograms can be clicked and the *values* field shows the exact number of occurrences in the interval clicked.

The x-values of the bars are the lower bounds of the interval represented by the bar. Histograms only display x-values with a non-zero y-value.

5.5 Trend curves

There are 8 trend curves.

SumPhilBud	The sum of philanthropists' budgets per iteration
SumBegBud	The sum of beggars' budgets per iteration
SumBegGra	The sum of beggars' grants per iteration
NumOfBeg	The number of surviving beggars after an iteration
NumMovBeg	The number of moving beggars per iteration
NumCollBeg	The number of collapsing beggars per iteration
NumRevBeg	The number of reviewed beggars per iteration
numOfCatas	The number of catastrophes per iteration (cf. section 3.2). Works only if the catastrophe alarm is set to True



Number of beggars

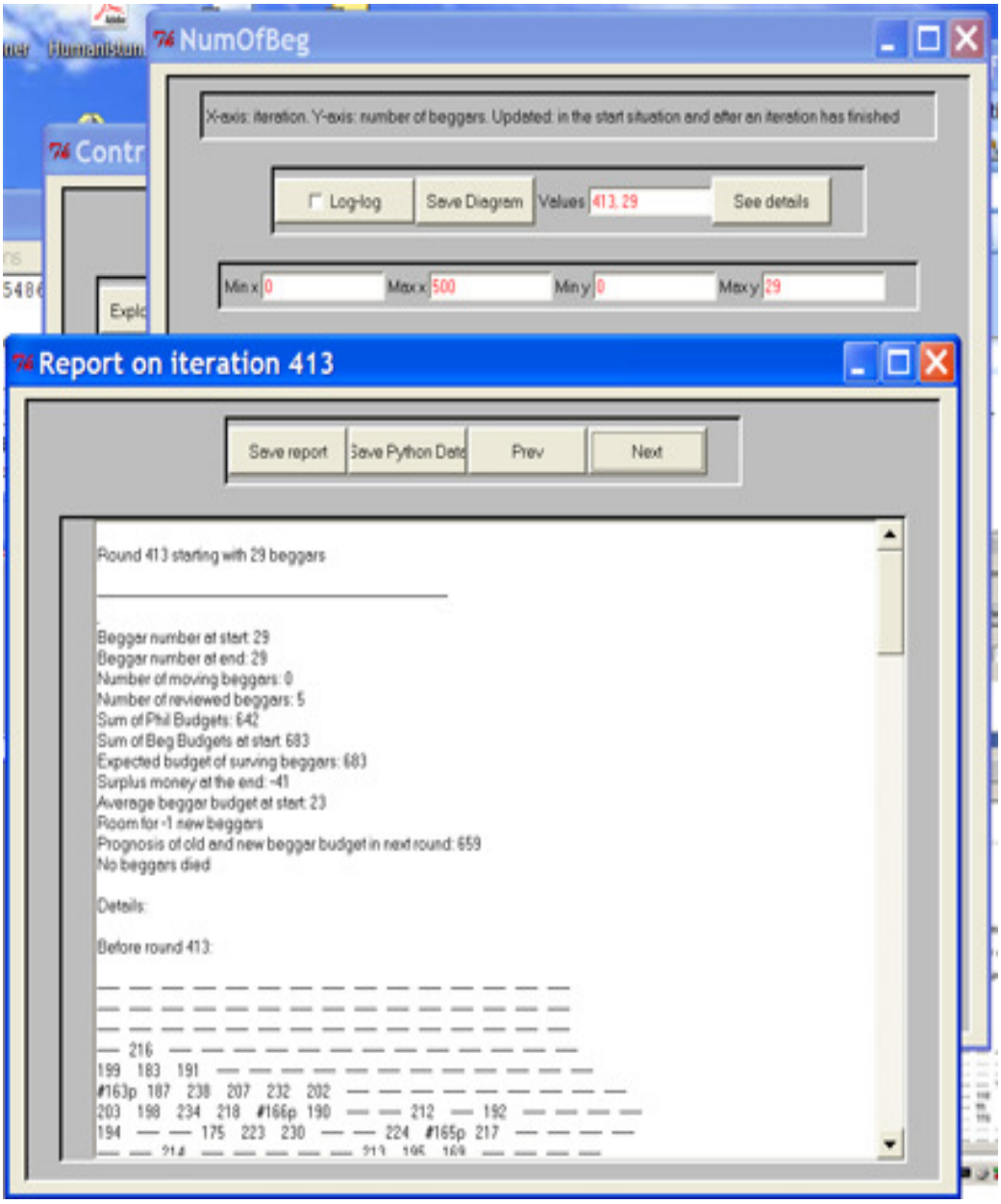
All trend curves have the iteration number on the x-axis and thus represent a development in time. It is possible to save the diagram as a postscript file, but it does not work on my machine.

It is possible to convert the diagram to a log/log plot, provided that it only contains positive values.

The four boxes display the minimum and maximum x and y-value with all decimals.

The trend-curves can be clicked and show the exact value recorded in the iteration clicked.

If one wants to do a more detailed study of an interesting part of the curve, click the part of the curve and click the *See details* button. Then a textual window describing the iteration pops up.


NumOfBeg
X-axis: iteration. Y-axis: number of beggars. Updated: in the start situation and after an iteration has finished
Log-log
Save Diagram
Values 413, 29
See details
Min x 0
Max x 500
Min y 0
Max y 29
Report on iteration 413
Save report
Save Python Data
Prev
Next
Round 413 starting with 29 beggars
Beggar number at start: 29
Beggar number at end: 29
Number of moving beggars: 0
Number of reviewed beggars: 5
Sum of Phil Budgets: 642
Sum of Beg Budgets at start: 683
Expected budget of surving beggars: 683
Surplus money at the end: -41
Average beggar budget at start: 23
Room for -1 new beggars
Prognosis of old and new beggar budget in next round: 659
No beggars died
Details:
Before round 413:
Detailed text windows

The text window has buttons for moving to the previous and next iteration. In this way, an interesting part of the curve may be studied in details. Note that much of the statistics are collected after an iteration has finished. It is the end result. Thus, at decrease of beggars is

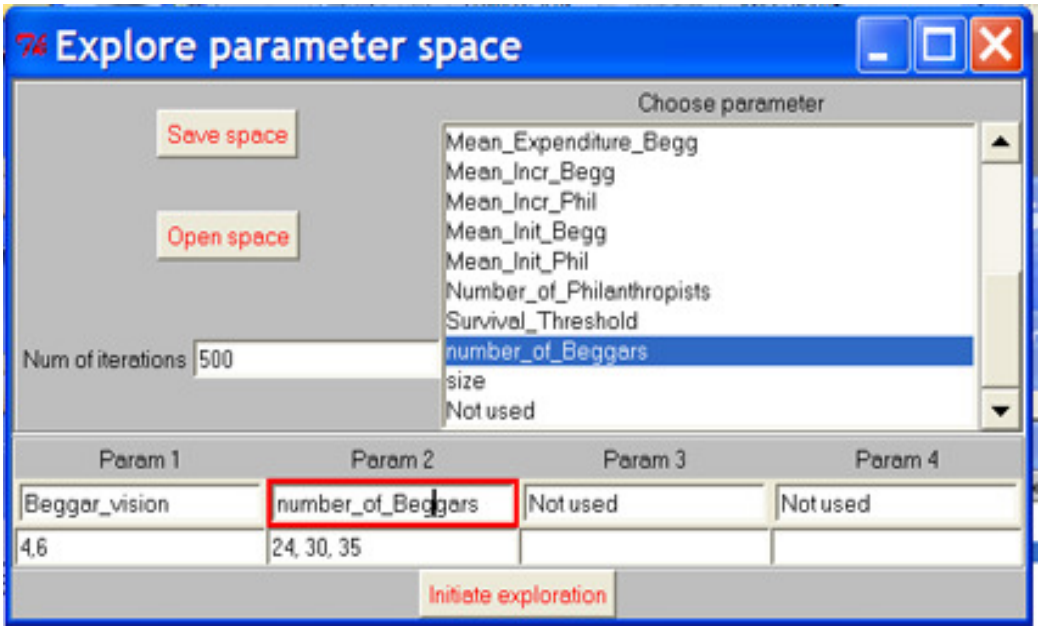
recorded in the iteration in which it occurs. If one is interested in the reason for beggar collapse, it is the low part of the curve that must be clicked.

6 Parameter space exploration

The system supports parameter space exploration. The idea is that you want to investigate what a particular set of parameters means for the system's behavior. What happens if we vary the number of beggars between 24, 30 and 35, and the beggars' range of vision between 4 and 6? Will the stability properties of the system change? Will we get a leptokurtic distribution for all combinations?

This part of the system is focused on producing a large file of comma-delimited beggar-states (Section 7.2) that can be input to a statistical package.

If you press the 'Explore param' button in the control window (section 5.1), the following window appears:



The window for editing parameter space exploration

The four text-widgets called 'Param1'…'Param4' can be clicked and brought into focus (see the red border above). If the listbox 'Choose parameter' is double-clicked, the parameter name is entered into the focused textbox (do not write in the textbox yourself, you risk making spelling errors!). The user can then enter up to four comma-separated values, which the parameter should take on, into the textbox *below* the focused textbox. For example in the figure, the user wants to explore a beggar vision of 4 and 6, combined with a beggar number of 24, 30, and 35.

The number of iterations can also be changed in the 'Num of iterations' box. Type the number followed by a RETURN.

The output of the parameter exploration is a comma-separated file containing beggar states for all the generated simulations. Each simulation is identified by a unique number.

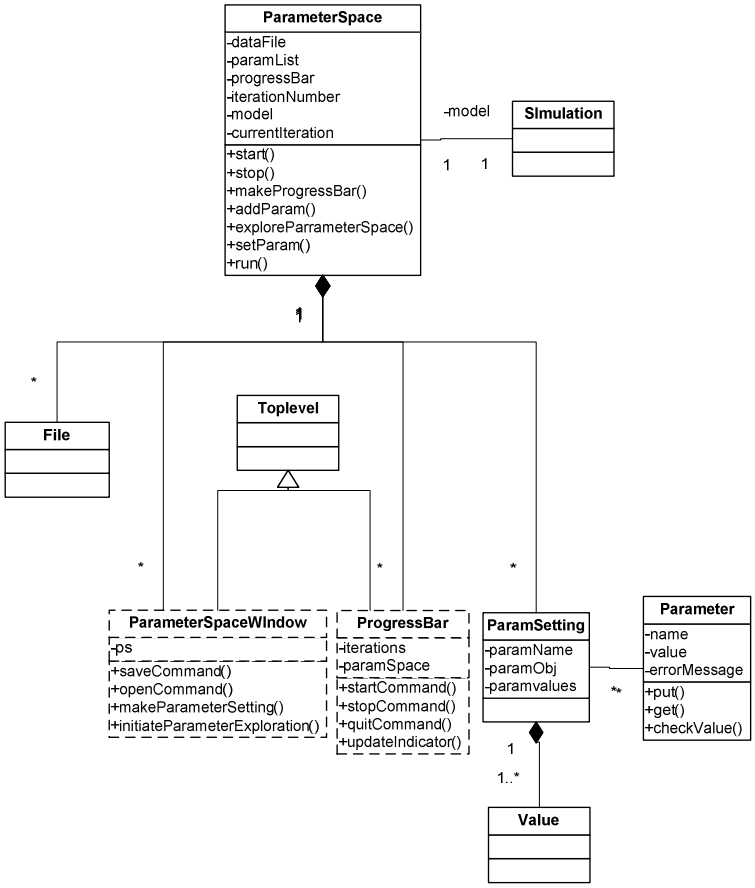
When the user has finished preparing the exploration of the parameter space, the button 'Initiate exploration' is clicked. Then the Progress Bar appears.

Fig. xxx. Progress bar for running of the parameter exploration

The Progress Bar calculates an estimate of the total number of iterations. The number of iterations is calculated as follows:

```
The number of parameters × the number of values of each parameter × the
number of iterations of each simulation
```

If 1000 iterations per simulation are selected, and all 4 parameters are used with 4 values, then the estimation will amount to 4 x 4 x 1000 = 16.000 iterations in total. If each iteration contains 24 beggars in average, there will be 16.000 x 24 = 380.000 measurements of beggars in total. The file may grow quite large. Therefore it is possible to 'Quit' the parameter exploration, if the data magnitude is too large. The parameter exploration can also be stopped. In this case, the current data are still kept on the file.



UML representation of the parameter space model

With a large parameter space it is a good idea to switch off the statistics check-boxes in the control window.

The parameter exploration part can be diagrammed as shown above. A parameter space contains zero or more parameter settings, each containing the parameter name, the parameter object and the parameter values. The parameter objects are systematically set all the values, and all combinations of values are simulated. The system simply does automatically what you can do yourself manually in the parameter-window. The combination of values is handled by the method *setParam* and the *run*-method takes care of the actual running of the simulation.

7 Files

Three main types of files can be saved to disc by means of the buttons *Save exceldata, Save Beggars*, and *Save Report Data.*

7.1 Save exceldata

The button produces 4 comma delimited files.

1.Statistics: 7 columns with values collected after each iteration. *SumPhilBud*: sum of budgets of philanthropists, *SumBegBud*: sum of beggars budgets, *SumBegGra*: sum of beggars grants, *NumOfBeg*: number of beggars after the iteration has finished, *NumMovBeg*: number of moving beggars during the iteration, *NumCollBeg*: number of collapsing beggars during the iteration, *NumRevBeg*: number of reviewed beggars during the iteration, *numOfCatas*: the number of catastrophes.

Example:

```
"SumPhilBud", "SumBegBud", "SumBegGra", "NumOfBeg", "NumMovBeg", "NumColl-
Beg", "NumRevBeg", "numOfCatas"
285, 479, 279, 13, 17, 11, 4, 0
284, 278, 269, 13, 1, 1, 1, 0
283, 275, 270, 14, 1, 0, 2, 0
277, 276, 269, 13, 0, 1, 2, 0
278, 276, 269, 13, 1, 1, 2, 0
277, 282, 258, 13, 0, 1, 0, 0
273, 261, 259, 13, 0, 0, 2, 0
269, 259, 257, 13, 0, 0, 2, 0
271, 262, 257, 13, 0, 0, 1, 0
272, 261, 254, 13, 0, 0, 1, 0
268, 260, 256, 13, 0, 0, 1, 0
266, 260, 253, 13, 0, 0, 1, 0
266, 262, 256, 13, 0, 0, 2, 0
269, 260, 256, 13, 0, 0, 2, 0
267, 261, 254, 13, 0, 0, 2, 0
271, 262, 256, 13, 0, 0, 2, 0
278, 264, 262, 13, 0, 0, 1, 0
283, 263, 263, 13, 0, 0, 0, 0
```

2. Lifespan: two columns, *value* and *number*. They indicate how many beggars (*number*) lived how long (*value*). The *value* represents lifespans ranging from *value* to but not including *value* + *step*, where *step* is the size of the interval. Here it is 2.

Example:

```
"value","number"
0, 5
2, 1
4, 1
6, 0
8, 0
```

```
10, 0
12, 0
14, 0
16, 0
18, 1
20, 0
22, 0
24, 0
26, 0
28, 0
30, 0
32, 0
34, 0
36, 0
38, 0
40, 0
42, 0
```

3. Grant Differences: two columns, value and number. They indicate how many beggars (*number*) experiences a certain grant difference from the previous iteration (*value*). The *value* denote all grant differences from *value* to but not including *value + step*, where *step* is the size of the interval. Here it is 2.

Example:

```
"value","number"
-39, 0
-37, 0
-35, 0
-33, 0
-31, 0
-29, 0
-27, 0
-25, 1
-23, 0
-21, 5
-19, 0
-17, 0
-15, 2
-13, 0
-11, 3
-9, 10
-7, 14
-5, 21
-3, 128
-1, 1029
1, 123
3, 24
5, 14
```

4. Applicant: two columns *value* and *number*. The *value* represents philanthropist budgets ranging from *value* to but not including *value + step*, where *step* is the size of the interval. *Number* is number of beggars that have applied to a philanthropist with a budget in that interval.

```
"value","number"
20, 0
30, 20
40, 152
50, 295
60, 32
70, 60
80, 156
90, 396
100, 145
110, 115
120, 10
```

7.2 Save beggars

The button produces 1 comma delimited file containing beggar states:

1 *beggarname*: name of a beggar
2 *iteration*: iteration number
3 *budget*: the beggar's budget in that iteration
4 *grant*: the beggars grant in that iteration
5 *distance*: the distance to the chosen philanthropist
6 *arrivals*: the number of beggars that have moved to the chosen philanthropist including the beggar himself
7 *departures*: the number of beggars that have moved from the chosen philanthropist including the beggar himself
8 *selfMoved*: 1 = the beggar has moved, 0 = he has not
9 *selfReviewed*: 1 = the beggar was reviewed, 0 = he was not
10 *appReviewed*: the number of applicants reviewed by the chosen philanthropist
11 *numOfComp*: the number of competitors seen by the beggar
12 *catastrophe*: 1 = the beggar experienced a catastrophe, 0 = he did not
13 *CoApplicants*: the actual number applicants to the chosen philanthropist
14 *deaths*: the number of coapplicants that went bankrupt
15 *simulationNo*: the simulation id

Example:

```
"beggarname", "iteration", "budget", "grant", "distance", "arrivals", "de-
partures", "selfMoved", "selfReviewed", "appReviewed", "numOfComp", "catas-
trophe", "coApplicants", "deaths", "simulationNo"
5, 1, 24.2482820501, 24.2482820501, 2, 5, 0, 1, 0, 1, 7, 0, 5, 1, 2
5, 2, 25.1341906558, 25.1341906558, 0, 0, 0, 0, 0, 0, 4, 0, 4, 0, 2
5, 3, 24.1381693649, 24.1381693649, 0, 0, 0, 0, 0, 1, 4, 0, 4, 0, 2
5, 4, 24.8238170611, 24.8238170611, 0, 0, 0, 0, 0, 1, 3, 0, 4, 0, 2
5, 5, 24.4457486073, 23.4094309164, 0, 0, 0, 0, 1, 1, 3, 0, 4, 0, 2
5, 6, 23.4123216612, 23.4123216612, 0, 0, 0, 0, 0, 0, 3, 0, 4, 0, 2
5, 7, 24.1959759357, 23.9804987022, 0, 0, 0, 0, 1, 1, 3, 0, 4, 0, 2
5, 8, 23.6113948208, 23.3068218144, 0, 0, 0, 0, 1, 1, 3, 0, 4, 0, 2
5, 9, 23.799136745, 23.799136745, 0, 0, 0, 0, 0, 0, 3, 0, 4, 0, 2
5, 10, 23.1118950195, 23.1118950195, 0, 0, 0, 0, 0, 0, 3, 0, 4, 0, 2
5, 11, 23.1356947129, 23.1356947129, 0, 0, 0, 0, 0, 0, 3, 0, 4, 0, 2
5, 12, 22.5945385503, 22.5945385503, 0, 0, 0, 0, 0, 0, 3, 0, 4, 0, 2
5, 13, 23.8205839219, 23.8205839219, 0, 0, 0, 0, 0, 1, 3, 0, 4, 0, 2
5, 14, 22.7038653015, 22.7038653015, 0, 0, 0, 0, 0, 1, 3, 0, 4, 0, 2
```

Beggar state files must conform to the following restrictions:

If	Then
selfMoved = 1	Distance >0 and arrivals > 0
selfReviewed = 1	Grant < budget
selfMoved = 0	$CoApplicants_i = CoApplicants_{i-1} + arrivals_i - departures_i - deaths_{i-1}$
$castrophe_i = 1$	$grant_i < grant_{i-1}$ or $grant_i > grant_{i-1}$
appReviewed = 0	selfReviewed = 0

If the parameter *Quality_assurance* in the parameter window is set to true, the system will check that the data conforms to these restrictions. *Quality_assurance* is set to false by default.

We can use these numbers to define interesting situations:

- *numOfComp <> CoApplicants:* this is a situation where the beggar's estimation of the situation does not agree with the factual situation. numOfComp > CoApplicants: a number of perceived competitors have fled after the beggar made his decision. numOfComp < CoApplicants: a number of new competitors have arrived after the beggar made his decision.

7.3 Save Reportdata

The button saves 1 textual file.

Report. Contains a detailed textual description of the simulation. There are the following sections:

1. parameter settings.

```
Parameters were set as follows:
--------------------------------
```

Beggars and Philantropists, version: beta 7, Time: 29-5-2008-16.28

Parameters were set as follows:
---

size = 15
Number_of_Beggars = 24
Number_of_Philanthropists = 4
Phil_vision = 4
Beggar_vision = 4
Iterations = 100
Survival_Threshold = 0.05
Mean_Init_Begg = 19
Mean_Incr_Begg = 0.001
Deviation_Incr_Begg = 0.02
Mean_Expenditure_Begg = 0.95
Deviation_Expenditure_Begg = 0.02
Mean_Init_Phil = 80
Deviation_Init_Phil = 20
Mean_Incr_Phil = 0.001
Deviation_Incr_Phil = 0.02
Mean_Expenditure_Begg = 0.95
Deviation_Expenditure_Begg = 0.02
Slowness_of_birth_(sec.) = 0.05
Slowness_of movement_(sec.) = 0.005
Beggar/Phil_ratio = 6
Base_Value = 0.25
Collect_stat_from = 5
Catastrophe_Alarm = False
Alarm_Sound = False
Alarm_set_point = 4

2. End result of simulation

End result of simulation:

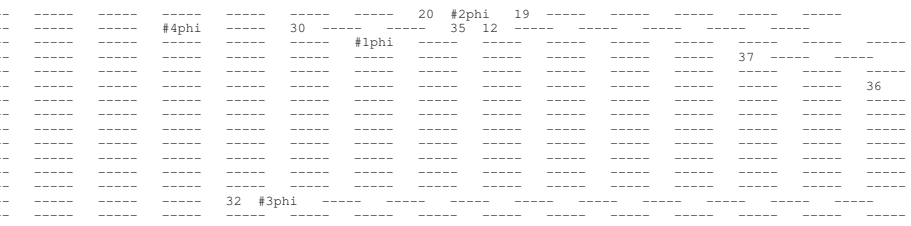
SumBegBud	NumRevBeg	NumMovBeg	Iteration	NumCollBeg	NumOfBeg	SumPhilBud	SumBegGra
613	18	24	1	16	8	239	231
209	2	4	2	1	7	246	185
214	2	4	3	2	6	253	171
198	0	4	4	0	7	258	198
258	3	3	5	3	6	262	210
182	0	1	6	0	6	270	182
221	0	3	7	1	7	270	221
256	1	3	8	2	6	275	233
200	0	2	9	0	6	279	200
238	2	3	10	3	5	289	204
226	1	3	11	1	5	295	198
174	0	2	12	1	5	302	174
219	1	3	13	3	5	308	202

……..

4. reports on each round:

Round 10 starting with 8 beggars

Before round 10:



4 philantropists: : #1phil, #2phil, #3phil, #4phil,
8 beggars: : 12, 19, 20, 30, 32, 35, 36, 37,

The following events happened:

12s life:
Iteration 10, found 4 philantropists, calculated 0(#4phil), 1(#3phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 4 philantropists - 6(#4phil), 12(#3phil) and 3(#1phil), calculated the successrates 1(#4phil), 1(#3phil), 1(#1phil) and 1(#2phil), selected philantropist #4phil, calculated a budget proposal of 34 crowns, moved to #4phil, sent proposal for 34 crowns to #4phil, received 34 crowns from #4phil, handled answer, spent 33 crowns

19s life:
Iteration 10, found 4 philantropists, calculated 1(#4phil), 1(#3phil), 1(#1phil) and 3(#2phil) competitors, calculated paths to 4 philantropists - 6(#4phil), 13(#3phil) and 3(#1phil), calculated the successrates 1(#4phil),

1(#3phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 34 crowns, sent proposal for 34 crowns to #2phil, received 34 crowns from #2phil, handled answer, spent 34 crowns

20s life:
 Iteration 10, found 4 philantropists, calculated 1(#4phil), 1(#3phil), 1(#1phil) and 3(#2phil) competitors, calculated paths to 4 philantropists - 4(#4phil), 13(#3phil) and 2(#1phil), calculated the successrates 1(#4phil), 1(#3phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 28 crowns, sent proposal for 28 crowns to #2phil, received 28 crowns from #2phil, handled answer, spent 29 crowns

30s life:
 Iteration 10, found 3 philantropists, calculated 1(#4phil), 1(#1phil) and 3(#2phil) competitors, calculated paths to 3 philantropists - 2(#4phil) and 3(#2phil), calculated the successrates 1(#4phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 43 crowns, moved to #2phil, sent proposal for 43 crowns to #2phil, received 43 crowns from #2phil, handled answer, spent 44 crowns

32s life:
 Iteration 10, found 1 philantropists, calculated 1(#3phil) competitors, calculated paths to 1 philantropists - , calculated the successrates 1(#3phil), selected philantropist #3phil, calculated a budget proposal of 33 crowns, sent proposal for 33 crowns to #3phil, received 33 crowns from #3phil, handled answer, spent 31 crowns

35s life:
 Iteration 10, found 2 philantropists, calculated 1(#1phil) and 4(#2phil) competitors, calculated paths to 2 philantropists - 2(#1phil), calculated the successrates 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 31 crowns, sent proposal for 31 crowns to #2phil, received 15 crowns from #2phil, handled answer, died because I needed to spend 32 and got 15, went bankrupt at 10 iteration

36s life:
 Iteration 10, found 0 philantropists, handled answer, went bankrupt at 10 iteration, died because of empty inbox

37s life:
 Iteration 10, found 1 philantropists, calculated 4(#2phil) competitors, calculated paths to 1 philantropists - 5(#2phil), calculated the successrates 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 33 crowns, moved to #2phil, sent proposal for 33 crowns to #2phil, received 16 crowns from #2phil, handled answer, died because I needed to spend 29 and got 16, went bankrupt at 10 iteration

#1phils life:
 Iteration 10, Replenished budget. New budget = 53, received no applications

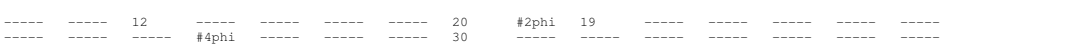
#2phils life:
 Iteration 10, Replenished budget. New budget = 138, received proposal for 34 crowns from 19, received proposal for 28 crowns from 20, received proposal for 43 crowns from 30, received proposal for 31 crowns from 35, received proposal for 33 crowns from 37, #2phil has succesrate 1, total of received applications was 170, reviewed 2 applications, and cut down with 0.510214526897, sent 34 crowns to 19, sent 28 crowns to 20, sent 43 crowns to 30, sent 15 crowns to 35, sent 16 crowns to 37

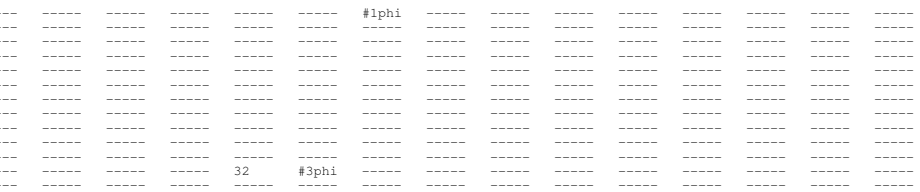
#3phils life:
 Iteration 10, Replenished budget. New budget = 51, received proposal for 33 crowns from 32, #3phil has succesrate 1, total of received applications was 33, granted all applications, sent 33 crowns to 32

#4phils life:
 Iteration 10, Replenished budget. New budget = 44, received proposal for 34 crowns from 12, #4phil has succesrate 1, total of received applications was 34, granted all applications, sent 34 crowns to 12

The causes of death was:
 1 died because of empty inbox
 2 died because of insufficent grant

After round 10:






```
4 philantropists: : #1phil, #2phil, #3phil, #4phil,
5 beggars: : 12, 19, 20, 30, 32,

1 new beggars generated
```

5. The individual lives:

```
Report on 75
--------------------
 1: 75 was born at iteration 29
 2: Iteration 30, found 3 philantropists, calculated 1(#4phil), 2(#1phil)
and 3(#2phil) competitors, calculated paths to 3 philantropists -
2(#4phil), 1(#1phil) and 3(#2phil), calculated the successrates
0.578763750486(#4phil), 0.793304632087(#1phil) and 1(#2phil), selected phi-
lantropist #2phil, calculated a budget proposal of 64 crowns, moved to
#2phil, sent proposal for 64 crowns to #2phil, received 64 crowns from
#2phil, handled answer, spent 58 crowns
 3: Iteration 31, found 2 philantropists, calculated 2(#1phil) and
3(#2phil) competitors, calculated paths to 2 philantropists - 2(#1phil),
calculated the successrates 0.793304632087(#1phil) and
0.968585636584(#2phil), selected philantropist #2phil, calculated a budget
proposal of 63 crowns, sent proposal for 63 crowns to #2phil, received 63
crowns from #2phil, handled answer, spent 66 crowns
 4: Iteration 32, found 2 philantropists, calculated 1(#1phil) and
3(#2phil) competitors, calculated paths to 2 philantropists - 2(#1phil),
calculated the successrates 1(#1phil) and 1(#2phil), selected philantropist
#2phil, calculated a budget proposal of 64 crowns, sent proposal for 64
crowns to #2phil, received 64 crowns from #2phil, handled answer, spent 61
crowns
 5: Iteration 33, found 2 philantropists, calculated 0(#1phil) and
3(#2phil) competitors, calculated paths to 2 philantropists - 2(#1phil),
calculated the successrates 1(#1phil) and 1(#2phil), selected philantropist
#1phil, calculated a budget proposal of 65 crowns, moved to #1phil, sent
proposal for 65 crowns to #1phil, received 65 crowns from #1phil, handled
answer, spent 66 crowns
 6: Iteration 34, found 3 philantropists, calculated 1(#4phil), 1(#1phil)
and 2(#2phil) competitors, calculated paths to 3 philantropists - 2(#4phil)
and 3(#2phil), calculated the successrates 1(#4phil), 1(#1phil) and
1(#2phil), selected philantropist #2phil, calculated a budget proposal of
67 crowns, moved to #2phil, sent proposal for 67 crowns to #2phil, received
67 crowns from #2phil, handled answer, spent 67 crowns
 7: Iteration 35, found 3 philantropists, calculated 1(#4phil), 1(#1phil)
and 3(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil)
and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and
1(#2phil), selected philantropist #2phil, calculated a budget proposal of
67 crowns, sent proposal for 67 crowns to #2phil, received 67 crowns from
#2phil, handled answer, spent 65 crowns
 8: Iteration 36, found 3 philantropists, calculated 1(#4phil), 1(#1phil)
and 3(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil)
and 1(#1phil), calculated the successrates 0.599559624617(#4phil),
1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget
proposal of 70 crowns, sent proposal for 70 crowns to #2phil, received 70
crowns from #2phil, handled answer, spent 65 crowns
 9: Iteration 37, found 3 philantropists, calculated 1(#4phil), 1(#1phil)
and 3(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil)
and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and
1(#2phil), selected philantropist #2phil, calculated a budget proposal of
71 crowns, sent proposal for 71 crowns to #2phil, received 71 crowns from
#2phil, handled answer, spent 69 crowns
 10: Iteration 38, found 3 philantropists, calculated 2(#4phil), 1(#1phil)
and 3(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil)
and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and
1(#2phil), selected philantropist #2phil, calculated a budget proposal of
73 crowns, sent proposal for 73 crowns to #2phil, received 73 crowns from
#2phil, handled answer, spent 73 crowns
 11: Iteration 39, found 3 philantropists, calculated 1(#4phil), 2(#1phil)
and 3(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil)
```

and 1(#1phil), calculated the successrates 0.655929347526(#4phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 75 crowns, sent proposal for 75 crowns to #2phil, received 75 crowns from #2phil, handled answer, spent 68 crowns
12: Iteration 40, found 3 philantropists, calculated 1(#4phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 75 crowns, sent proposal for 75 crowns to #2phil, received 75 crowns from #2phil, handled answer, spent 74 crowns
13: Iteration 41, found 3 philantropists, calculated 1(#4phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 79 crowns, sent proposal for 79 crowns to #2phil, received 79 crowns from #2phil, handled answer, spent 73 crowns
14: Iteration 42, found 3 philantropists, calculated 1(#4phil), 2(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 78 crowns, sent proposal for 78 crowns to #2phil, received 78 crowns from #2phil, handled answer, spent 76 crowns
15: Iteration 43, found 3 philantropists, calculated 1(#4phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 0.810783834722(#2phil), selected philantropist #2phil, calculated a budget proposal of 82 crowns, sent proposal for 82 crowns to #2phil, received 82 crowns from #2phil, handled answer, spent 75 crowns
16: Iteration 44, found 3 philantropists, calculated 1(#4phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 82 crowns, sent proposal for 82 crowns to #2phil, received 82 crowns from #2phil, handled answer, spent 83 crowns
17: Iteration 45, found 3 philantropists, calculated 1(#4phil), 1(#1phil) and 3(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 0.78968739956(#2phil), selected philantropist #2phil, calculated a budget proposal of 85 crowns, sent proposal for 85 crowns to #2phil, received 85 crowns from #2phil, handled answer, spent 76 crowns
18: Iteration 46, found 3 philantropists, calculated 1(#4phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 86 crowns, sent proposal for 86 crowns to #2phil, received 86 crowns from #2phil, handled answer, spent 80 crowns
19: Iteration 47, found 3 philantropists, calculated 1(#4phil), 2(#1phil) and 3(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 1(#2phil), selected philantropist #2phil, calculated a budget proposal of 87 crowns, sent proposal for 87 crowns to #2phil, received 87 crowns from #2phil, handled answer, spent 86 crowns
20: Iteration 48, found 3 philantropists, calculated 2(#4phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 0.956571125306(#2phil), selected philantropist #2phil, calculated a budget proposal of 89 crowns, sent proposal for 89 crowns to #2phil, received 89 crowns from #2phil, handled answer, spent 89 crowns
21: Iteration 49, found 3 philantropists, calculated 2(#4phil), 2(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 0.708694164405(#4phil), 1(#1phil) and 0.956550997146(#2phil), selected philantropist #2phil, calculated a budget proposal of 92 crowns, sent proposal for 92 crowns to #2phil, received 43 crowns from #2phil, handled answer, spent 91 crowns
22: Iteration 50, found 3 philantropists, calculated 2(#4phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 1(#4phil), 1(#1phil) and 0.779362069208(#2phil), selected philantropist #2phil, calculated a budget proposal of 95 crowns, sent proposal for 95 crowns to #2phil, received 95 crowns from #2phil, handled answer, spent 92 crowns
23: Iteration 51, found 3 philantropists, calculated 1(#4phil), 1(#1phil) and 4(#2phil) competitors, calculated paths to 3 philantropists - 4(#4phil) and 1(#1phil), calculated the successrates 0.723605180976(#4phil), 1(#1phil) and 0.959465366254(#2phil), selected philantropist #2phil, calculated a budget proposal of 97 crowns, sent proposal for 97 crowns to #2phil, received 86 crowns from #2phil, handled answer, died because I needed to spend 100 and got 86, went bankrupt at 51 iteration

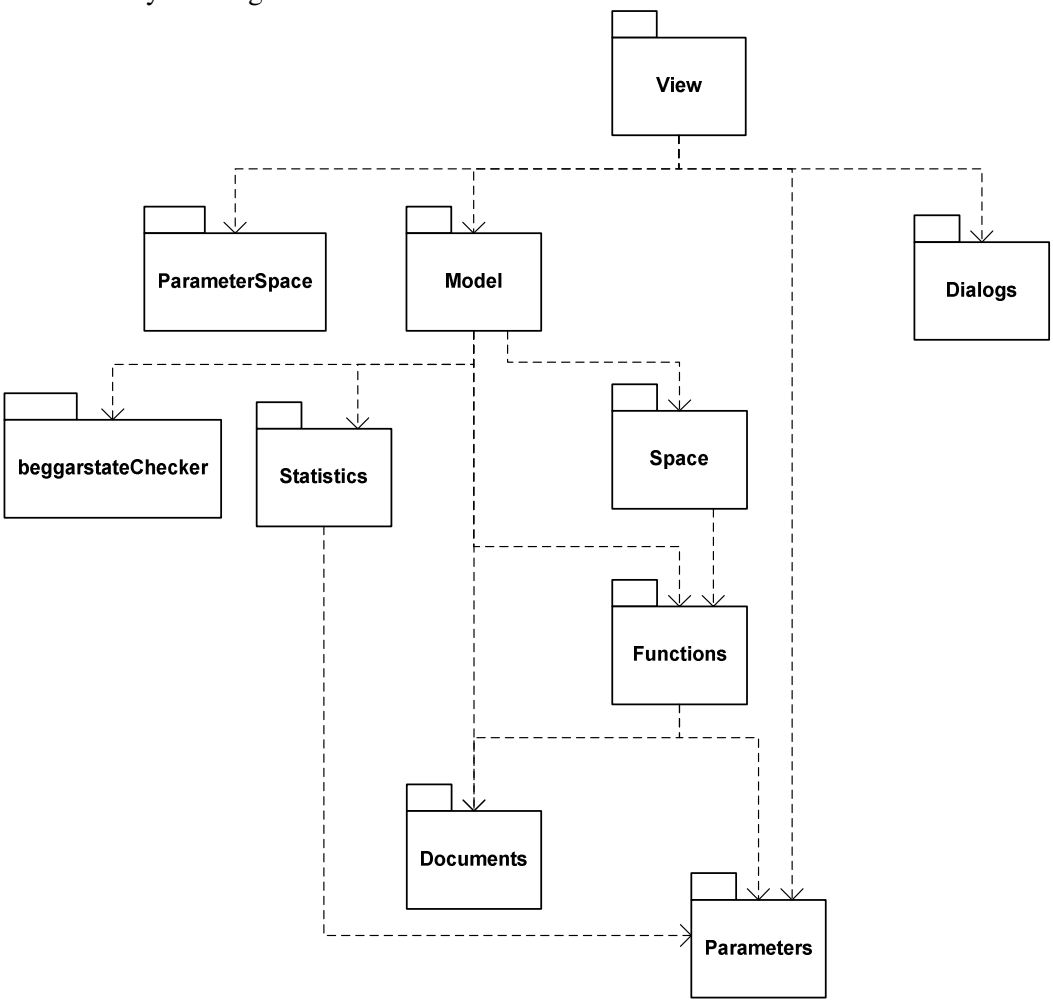
Report on #3phil

1: #3phil was born at iteration 0
2: Iteration 1, Replenished budget. New budget = 40, received proposal for 25 crowns from 10, received proposal for 25 crowns from 11, received proposal for 26 crowns from 15, #3phil has succesrate 0, total of received applications was 77, reviewed 2 applications, and cut down with 0.295347844001, sent 25 crowns to 10, sent 7 crowns to 15, sent 7 crowns to 11
3: Iteration 2, Replenished budget. New budget = 42, received proposal for 27 crowns from 20, #3phil has succesrate 0.513191798001, total of received applications was 27, granted all applications, sent 27 crowns to 20
4: Iteration 3, Replenished budget. New budget = 44, received proposal for 26 crowns from 24, #3phil has succesrate 1, total of received applications was 26, granted all applications, sent 26 crowns to 24
5: Iteration 4, Replenished budget. New budget = 43, received proposal for 26 crowns from 26, #3phil has succesrate 1, total of received applications was 26, granted all applications, sent 26 crowns to 26
6: Iteration 5, Replenished budget. New budget = 44, received proposal for 28 crowns from 26, received proposal for 27 crowns from 32, #3phil has succesrate 1, total of received applications was 56, reviewed 1 applications, and cut down with 0.57761817749, sent 27 crowns to 32, sent 16 crowns to 26
7: Iteration 6, Replenished budget. New budget = 47, received proposal for 29 crowns from 32, #3phil has succesrate 0.783088474632, total of received applications was 29, granted all applications, sent 29 crowns to 32
8: Iteration 7, Replenished budget. New budget = 46, received proposal for 32 crowns from 32, #3phil has succesrate 1, total of received applications was 32, granted all applications, sent 32 crowns to 32
9: Iteration 8, Replenished budget. New budget = 47, received proposal for 32 crowns from 32, #3phil has succesrate 1, total of received applications was 32, granted all applications, sent 32 crowns to 32
10: Iteration 9, Replenished budget. New budget = 49, received proposal for 34 crowns from 32, #3phil has succesrate 1, total of received applications was 34, granted all applications, sent 34 crowns to 32
11: Iteration 10, Replenished budget. New budget = 51, received proposal for 33 crowns from 32, #3phil has succesrate 1, total of received applications was 33, granted all applications, sent 33 crowns to 32
12: Iteration 11, Replenished budget. New budget = 53, received proposal for 33 crowns from 32, received proposal for 47 crowns from 38, #3phil has succesrate 1, total of received applications was 81, reviewed 1 applications, and cut down with 0.403545972249, sent 33 crowns to 32, sent 19 crowns to 38
13: Iteration 12, Replenished budget. New budget = 54, received proposal for 32 crowns from 32, #3phil has succesrate 0.64748119469, total of received applications was 32, granted all applications, sent 32 crowns to 32
14: Iteration 13, Replenished budget. New budget = 55, received proposal for 35 crowns from 32, received proposal for 36 crowns from 40, #3phil has succesrate 1, total of received applications was 71, reviewed 1 applications, and cut down with 0.561454233463, sent 35 crowns to 32, sent 20 crowns to 40
15: Iteration 14, Replenished budget. New budget = 56, received proposal for 37 crowns from 32, #3phil has succesrate 0.773693574859, total of received applications was 37, granted all applications, sent 37 crowns to 32
16: Iteration 15, Replenished budget. New budget = 57, received proposal for 37 crowns from 32, #3phil has succesrate 1, total of received applications was 37, granted all applications, sent 37 crowns to 32
17: Iteration 16, Replenished budget. New budget = 57, received proposal for 38 crowns from 32, #3phil has succesrate 1, total of received applications was 38, granted all applications, sent 38 crowns to 32
18: Iteration 17, Replenished budget. New budget = 59, received proposal for 38 crowns from 32, received proposal for 39 crowns from 48, received proposal for 37 crowns from 49, received proposal for 38 crowns from 50, #3phil has succesrate 1, total of received applications was 154, reviewed 3 applications, and cut down with 0.165331508363, sent 38 crowns to 32, sent 38 crowns to 50, sent 6 crowns to 48, sent 0 crowns to 49, sent 0 crowns to 49
19: Iteration 18, Replenished budget. New budget = 61, received proposal for 38 crowns from 32, received proposal for 36 crowns from 50, received proposal for 54 crowns from 51, #3phil has succesrate 0.543379475673, total of received applications was 129, reviewed 2 applications, and cut down with 0.251326836244, sent 38 crowns to 32, sent 9 crowns to 50, sent 13 crowns to 51
20: Iteration 19, Replenished budget. New budget = 61, received proposal for 37 crowns from 32, #3phil has succesrate 0.470981001056, total of received applications was 37, granted all applications, sent 37 crowns to 32

```
 21: Iteration 20, Replenished budget. New budget = 61, received proposal
for 41 crowns from 32, #3phil has succesrate 1, total of received applica-
tions was 41, granted all applications, sent 41 crowns to 32
 22: Iteration 21, Replenished budget. New budget = 60, received proposal
for 43 crowns from 32, #3phil has succesrate 1, total of received applica-
tions was 43, granted all applications, sent 43 crowns to 32
 23: Iteration 22, Replenished budget. New budget = 62, received proposal
for 43 crowns from 32, received proposal for 52 crowns from 61, #3phil has
succesrate 1, total of received applications was 96, reviewed 1 applica-
tions, and cut down with 0.213980825665, sent 52 crowns to 61, sent 9
crowns to 32
 24: Iteration 23, Replenished budget. New budget = 64, received no appli-
cations
 25: Iteration 24, Replenished budget. New budget = 65, received proposal
for 52 crowns from 19, received proposal for 45 crowns from 65, #3phil has
succesrate 0.638281218745, total of received applications was 98, reviewed
1 applications, and cut down with 0.362890094534, sent 45 crowns to 65,
sent 19 crowns to 19
```

8 Modules

There are 10 modules organized in a layered structure. If an arrow leads from A to B, A uses B but not conversely. This means that all modules can run individually. The whole system is run by clicking View.



UML package diagram. The modules of the system.

9 References

Prange, F. & S. Serritzlew (2005). A Game of Beggars and Philanthropists. A Dynamic Model of Stability and Punctuations in Budgeting, vers. 3.0.. University of Aarhus/ ProjectComplexity.

Prange, F. (2006). On playing non-equilibrium games: modeling the complex dynamics of games. *Ecological Complexity,* 3, 2006.